# SPECIFICATION OF DISTRIBUTED INFORMATION SYSTEMS

EDITORS
Hans-Dieter Ehrich
Ursula Goltz
José Meseguer
Amir Pnueli

# Contents

## 1 Preface

The specification, development and use of distributed information systems is an important research area with many practical applications. Examples for existing such systems are those owned by banks, airlines or governments. Moreover, the explosive growth of the Internet is evidence that a global information structure is developing, and expansion in the use of large distributed information systems can be predicted.

Information systems are *reactive* systems. Unlike transformational systems—that is, systems whose function is to transform some inputs into output results—reactive systems have an unlimited number of interactions with their environment. Basic notions of program correctness do not apply, and techniques for designing transformational systems do not easily carry over to reactive systems. Non-distributed reactive systems, like conventional operating and database systems, are not easy to design, implement and restructure, a lot remains to be done to develop helpful high-level concepts, features, languages, methods and tools. However, existing products can already cope successfully with reasonably complex systems in practice. Formal methods from logic, algebra and process theory have been successfully applied, for instance in query and transaction processing, but they are not in general standard engineering practice yet.

*Distributed* reactive systems are still an order of magnitude more complex and difficult to specify, analyse, implement and reconfigure. They are not well understood, and methods for designing, implementing and reengineering them have not yet reached a level of mature and sound engineering practice. A great challenge is to cope with *concurrency, synchronization, and communication* at an appropriate level of abstraction. This can be achieved by applying concepts from concurrency theory for distributed information systems.

This seminar was the follow-up of the Dagstuhl Seminar 98071 "Information Systems as Reactive Systems", which took place in February 1998. Until that seminar, there was little interaction between the communities concerned with information systems and concurrency theory. One outcome of the seminar was that there is a definite need for interaction; new contacts have been established. The focus of the present seminar was again to bring experts from the respective fields together. In contrast to the previous seminar, we put even more focus on applications, seeing the design of distributed information systems not only as theoretically challenging but also as an engineering problem.

In particular, the following topics have been addressed:

- Modelling and design methods.
  Information systems involve data, objects and global processes. In conceptual modeling, these aspects have been addressed in isolation and various combinations, and the same holds for the conceptual modeling perspectives: system architecture, object classes, object behavior, and object communication. Diagrammatic modeling approaches like OOA&D, OMT, OOSE and their recent amalgamation UML employ a multitude of specification concepts and techniques that are not always well integrated and sometimes not well understood. Approaches for giving formal semantics for (subsets of) UML would give a more clear picture of the underlying system model.

  - Activity charts contain ambiguities and many fuzzy constructs like swimlanes. The formalization of activity charts in terms of clocked Kripke structures was discussed, for which ATCTL can be used as property specification language to do model checking.
  - UML sequence diagrams can be used to specify requirements. An approach was introduced in which sequence diagrams with timing contraints and a system modeled in Uppaal timed automata are checked for consistency. For this the sequence diagrams were translated in Uppaal automata and consistency is verified by the Uppaal model checker.
  - Validation of a formal specification is concerned with the question if a specification precisely reflects the informal requirements. One way to validate process specifications is the generation and visualisation of process nets from a system model given by a Petri net.
  - Whenever the visual means of a modeling language will change, a re-implementation of supporting tools is necessary. In order to avoid such re-implementation, a generator for graphic editors was presented. In this approach a visual language is specified by a visual grammar which is based on a visual alphabet.

- Semantics of object-oriented specification languages.
  Describing the semantics of a formal language amounts to giving a map from syntactic constructs into a suitable semantic domain. For object-oriented specification languages, possible domains are process algebras, Petri nets, event structures, the actor model, type algebra, and others. Studies on relating and evaluating such domains are necessary. Interesting language issues are (amongst others) module composition, refinement and atomicity.

  - The use of labelled transitions systems was considered for the purpose of specifying and describing the behaviour of object-based systems.
  - MSCs possess a standard semantics based on process algebra which is rather complicated. An alternative are the theory of traces and partially commutative monoids. The trace based semantics is bisimular with the operational semantics and allows to apply the rich theory of partially commutative monoids with associated algorithms and tools.

  - It is important that integrity and correctness of specifications on higher levels are not damaged by a translation by a compiler. It was shown how the result of a translation can be verified.

- Logics for distributed systems.
  An attractive model for the execution of a distributed system is a *distributed process*, which has a time line for each location of the system. Temporal logics based on this notion are particularly suited for the specification of distributed systems. Rewriting logic is another well known and quite successful paradigm in this field. There is a multi-purpose specification language, Maude, that is based on rewriting techniques. Mobile Maude is a mobile agent language extending Maude and supporting mobile computation. Mobile Maude uses reflection to obtain a simple and general declarative mobile language design and makes strong assurances of mobile agent behaviour.

  - The question was discussed how global conditions for object systems can be split and assigned to a group of objects such that these objects together satisfy the global condition. One approach is to give the global statements in a distributed logic which allows for local assertions about the states of other objects in a local logic.
  - A semantic model of both external and internal dynamics of agent systems was given which uses locality and temporal compatibility as basic concepts.
  - A semantic approach was proposed which makes precise the reflective concept of composable services (security, faulttolerance, performance) in a distributed system. The approach is based on the executable formal semantics for distributed object-oriented systems provided by rewriting logic.
  - To deal with open system specifications a talk uses tile logic, where both closed and open terms are managed analogously. This allows to analyse the *bisimularity as congruence* property for serveral tile formats that accomplish different concepts of subterm sharing.
  - The recent developments in the area of automatic verification of logics referring to the distribution of a system were summerized. These logics refer to the causal order of events happening in a distributed system, either referring to infinite partial order of executions or to the prefix ordering of finite partial order executions.

- Mobility.
  For modeling large systems of communicating objects, it is indispensible to deal with a dynamically changing communication structure. In process algebras, this issue has been formally dealt with in the $\pi$-calculus and, more recently, in the fusion calculus.

  – To verify temporal and functional properties cast in terms of the $\pi$-calculus, a $\pi$-$\mu$-calculus (extension of the modal $\mu$-calculus) was introduced as an appropriate temporal logic for the $\pi$ -calculus. Moreover a variant of the $\pi$-calculus, known as spi-calculus, was used for the specification of secure communication protocols and for verifying security properties by means of equivalence checking.

- Application-oriented concepts in specification languages.
  Among the issues to be addressed are real time, fault tolerance, and internet applications.

  – A profile for modeling multimedia applications was presented. The profile OMMMA-L (Object-oriented Modeling of Multimedia Application - the Language) introduces a so-called presentation diagram to model the spatial aspect (layout) of a multimedia application.
  – A framework was introduced which uses the codesign approach based on an abstraction layer model to develop information-intensive internet sites.
  – A talk discussed media as model to envision, to design and to formalize platforms for communities of collaborating human or artificial agents. In this approach the media concept envisions media as plattforms of multi-agent systems and the media reference model determines the main components of a medium and guides its application.
  – A new formalism for modeling real-time systems in different application areas and on various levels of abstraction was presented. For I/O interval structures in the model, efficient model checking and analysis techniques based on MTBDDs exist in this approach.
  – Information system objects are subject to frequently occurring modifications of behaviour rules. A proposal was discussed for extending object specification techniques with capabilities to supporting evolving behaviour descriptions. The semantics of these added language constructs can be expressed e. g. using an extended temporal logic framework.
  – The method Focus offers a formal framework for topdown-development of distributed, reactive systems. It was demonstrated how this method can be applied to the modeling of operating system concepts.

The programme of the seminar was intense and stimulating; it comprised 23 talks, the abstracts are recorded in this report in alphabetical order. Some of the basic principles of verification of concurrent systems are summarized in a new book which was presented at the workshop. We had an interesting panel discussion in which the panelists discussed the question if and how results of the concurrency theory can help to develop distributed information systems and which are the most promising aspects from which distributed information systems can benefit.

**Acknowledgements**

Many thanks to the Dagstuhl team and to Karsten Diethers and Ralf Pinger for their help in the preparation and organisation of the seminar. On behalf of all the participants, the organisers would also like to thank the staff of Schloß Dagstuhl for providing an excellent environment.

Hans-Dieter Ehrich
Ursula Goltz
José Meseguer
Amir Pnueli

## 2 Workshop Programme

**Monday 21**

*Chair: Ursula Goltz*

9.00 Ugo Montanari: Bisimilarity Congruences for Open Terms and Term Graphs via Tile Logic
9.45 Roel Wieringa: Issues in the Semantics of UML Activity Diagrams
10.30 COFFEE BREAK

*Chair: Hans-Dieter Ehrich*

11.00 Bernhard Thalheim: Specification and Development of Information Services for the Internet Cable Nets

12.10 LUNCH BREAK

*Chair: José Meseguer*

14.15 Arend Rensink: Labelled Transition Systems for Object System Behaviour
15.00 Ulrike Lechner: Media – A Model of Distributed Information Systems
15.45 COFFEE BREAK

*Chair: Mads Dam*

16.15 Peter Niebert: Model Checking of Causal and Distributed Logics

---

**Tuesday 22**

*Chair: Willem Paul de Roever*

9.00 José Meseguer: Mobile Maude
9.45 Rocco De Nicola: Proof Techniques for Cryptographic Processes
10.30 COFFEE BREAK

*Chair: Mads Dam*

11.00 Mogens Nielson: Dangers of the Past

12.10 LUNCH BREAK

*Chair: Mogens Nielson*

14.00 Willem Paul de Roever: Basic Principles of the Concurrency-Verification Book
15.00 Mads Dam: Proof Systems for $\pi$-Calculus Logics
15.45 COFFEE BREAK

*Chair: Roel Wieringa*

16.15 Katharina Spies: Formal Modeling of Operating Systems Concepts – A Methodical Approach

---

**Wednesday 23**

*Chair: Ugo Montanari*

9.00 Amir Pnueli: Translation Validation for Optimizing Compilers
9.45 Jörg Desel: Validation of Process Specifications

10.30 COFFEE BREAK

*Chair: Friedrich von Henke*

11.00 Grit Denker: Rewriting Semantics for Meta-Objects and Distributed Composable Services

12.10 LUNCH BREAK

14.00 EXCURSION

---

**Thursday 24**

*Chair: Rocco De Nicola*

9.00 Hans-Dieter Ehrich: Checking Object Systems
9.45 Gunter Saake: Evolving Object Specifications for Adaptable Systems

10.30 COFFEE BREAK

*Chair: Bernhard Thalheim*

11.00 Pascal van Eck: Semantic Formalization of Emerging Dynamics of Compositional Agent Systems

12.10 LUNCH BREAK

*Chair: Gunter Saake*

14.15 Jürgen Ruf: Modeling, Verification and Analysis of Networks of Real-Time Systems
15.00 Piotr Kosiuczenko: Message Sequence Charts form the Partially Monoids Point of View

15.45 COFFEE BREAK

*Chair: José Meseguer*

16.15 DISCUSSION

---

**Friday 25**

*Chair: Amir Pnueli*

9.00 Gregor Engels: Object-Oriented Modeling of Multimedia Applications
9.45 Roswitha Bardohl: Visual Specification of Diagrammatic Modeling Techniques

10.30 COFFEE BREAK

*Chair: Gregor Engels*

11.00 Karsten Diethers: Timed Sequence Diagrams and Tool-Based Analysis – A Case Study

12.10 LUNCH

END

# 3 Collected Abstracts

## Visual Specification of Diagrammatic Modeling Techniques

*Roswitha Bardohl*
TU Berlin
FB 13 Informatik

Nowadays numerous diagrammatic techniques and visual environments exist that can be used for the modeling or specification of software systems. Examples are given, e.g., by the Unified Modeling Language (UML) and supporting tools like Rational Rose. In general, each of such tools offer graphic editors for diagram drawing. In these editors, the visual means of the corresponding visual language are mostly tightly integrated. Whenever the visual means of the language will change, a time and cost intensive re-implementation is necessary. In order to avoid such re-implementations, generators for graphic editors or environments come into the fore. Usually, such generators expect a textual specification instead of a visual one. However, textual specifications are difficult to read because of the two-dimensional character of diagrammatic languages.

In this talk, concepts for the visual specification of visual languages are proposed as well as a corresponding visual environment which is called GenGEd. From a user-defined visual specification, GenGEd generates a graphic editor allowing for syntax aided editing. The visual languages that can be specified using GenGEd are diagrammatic languages commonly used for software modeling or specification.

Similar to formal textual languages, in this approach a visual language is specified by a visual grammar which is based on a visual alphabet. In contrast to textual languages where the symbols are placed one beside the other, the placement of visual symbols is already part of the visual alphabet. These concepts are based on the well-defined theory of algebraic graph transformation that do not restrict the visual grammars to context-free grammars. Moreover, the grammar rules are context-sensitive and in addition, they can be enhanced with some application conditions. Thus, this approach allows for specifying a broad spectrum of visual languages.

In accordance to the concepts for the visual specification of visual languages, the GenGEd specification tool comprises an alphabet editor and a grammar editor. The usage of these sequently toggled editors result in a visual language specification that is the input of a generic graphic editor. The editing commands are given by the grammar rules of the specification. The transformation of visual sentences, i.e., the editing of diagrams, is supported by the graph transformation machine Agg (developed at Technical University of Berlin) and the graphical constraint solver ParCon (developed at University of Paderborn).

The AGG language allows for the specification of software systems by graph means. The language is already extended by distribution concepts. These concepts are being implemented in the AGG environment. Because of using AGG in the GENGED environment, we are looking forward to an extended AGG version such that also GENGED can provide these features in order to support not only the specification of visual languages used for the specification of distributed systems but also their execution.

---

## Proof Systems for $\pi$-Calculus Logics

*Mads Dam*

Swedish Institute of Computer Science
Formal Design Techniques Laboratory

In this paper we study the problem of verifying general temporal and functional properties of mobile and dynamic process networks, cast in terms of the pi-calculus. Much of the expressive power of this calculus derives from the combination of name generation and communication (to handle mobility) with dynamic process creation. In the paper we introduce the $\pi$-$\mu$-calculus, an extension of the modal mu-calculus with name equality, inequality, first-order universal and existential quantification, and primitives for name input and output as an appropriate temporal logic for the pi-calculus. A compositional proof system is given with the scope of verifying dynamic networks of pi-calculus agents against properties specified in this logic. The proof system consists of a local part based, roughly, on the classical sequent calculus extended with data structures for private names, and rules to support process structure dependent reasoning. In addition the proof system contains a rule of discharge to close well-founded cycles in the proof graph. The proof system is shown to be sound in general and weakly complete for the non-recursive fragment of the specification logic. We also obtain a weak completeness result for recursive formulas against finite-control calculus processes. Two extended examples are considered. The first example is based on Milner's encoding of data types into the pi-calculus, specifically the natural numbers, This encoding is interesting from the point of view of verification, since it makes essential use of all the distinguishing features of the pi-calculus, including dynamic process creation. Corresponding to the encoding of natural numbers into the $\pi$-calculus we propose an encoding of the type of natural numbers into the $\pi$-$\mu$-calculus and establish some type correctness properties. As the second example we consider a garbage-collecting unbounded buffer (which dynamically create and destroy buffer cells) and show how to establish absence of spurious output of such a system.

---

## Proof Techniques for Cryptographic Processes

*Rocco De Nicola*
Universita di Firenze
Dipartimento di Sistemi e Informatica

We introduced a variant of the pi-calculus, known as spi-calculus, that permits exchange of crypted values and channel names. Then, we showed, via a few examples, the use of such calculus for the specification of secure communication protocols and for verifying security properties by means of equivalence checking. Finally, we presented our original contribution to the field amounting to a variation of two behavioral equivalences that renders them more amenable to automatic verification.

Classically, equivalences for cryptographic process calculi are defined by relying on universal quantifcation over the possible observational contexts and this significantly complicates the verification procedure. We focussed on two such equivalences, namely may-testing and barbed bisimulation equivalence, and investigated tractable proof methods for them. We have designed an enriched labelled transition system, where transitions are constrained by the knowledge the environment has of names and keys. The new transition system is used to define a trace equivalence and a weak bisimulation equivalence, that avoid quantification over contexts. Our main results are soundness and completeness of trace and weak bisimulation equivalence with respect to may-testing and barbed equivalence, respectively. They lead to more direct proof methods for equivalence checking. The use of these methods was illustrated with an example, concerning implementation of secure channels by means of encrypted public channels.

An extended abstract of the presented work appeared as: M. Boreale, R. De Nicola, R. Pugliese.Proof Techniques for Cryptographic Processes. Proc. of 14th Symposium on Logic in Computer Science (LICS'99), IEEE Computer Society Press, 1999.

The full version of the paper can be retrieved at
ftp://rap.dsi.unifi.it/pub/papers/BDP-spi.ps.gz.

## Rewriting Semantics for Meta-Objects and Distributed Composable Services

*Grit Denker*

SRI International
Computer Science Laboratory

In distributed computing and in communications software there is a great interest in modular and composable approaches. In particular, services such as security or fault-tolerance and services intended for boosting performance that may in practice be "hard-wired" across different parts of the code of a distributed system should be treated in a much more modular way, so that they can be dynamically added to a system, and so that several such services can be composed to obtain their combined benefits. Besides the good software engineering reasons of thus making the systems much more easily reusable and adaptable, modularity and composability are also crucial at runtime, in the sense that some of these services, that may have a cost in performance, should not be used always and everywhere. Instead, they should be installed dynamically and selectively at runtime, in those areas or domains of the distributed system where they are needed in response to some security threat, failure, need for boosting performance, and so on.

In this talk we propose a semantic approach to make precise the reflective concept of composable service in a distributed system, and to reason about the properties of service compositions. Our approach is based on the executable formal semantics for distributed object-oriented systems provided by rewriting logic and explicitly addresses the reflective properties that are essential for having a truly modular notion of service. The formal model that we present seems promising not only for formal analysis and symbolic simulation, but also for the application of theorem proving methods to formally verify important properties of services and their compositions. We have been strongly influenced by the onion skin model of actor reflection developed by Agha et al which we generalize and formalize within the framework of rewriting logic.

Joint work with Jose Meseguer (SRI International) and Carolyn Talcott (Stanford University).

## Validation of Process Specifications

*Jörg Desel*
Katholische Universität Eichstätt
Lehrstuhl für Angewandte Informatik

Verification is concerned with the correctness of a program or, more abstractly, of a system model with respect to a given specification. But even if a model can be proven to be correct in this sense, it might exhibit a behavior which does not correspond to given restrictions, requirements, plans or ideas. In other words, even if the system model is right, it is not necessarily the right system model. Validation of a formal specification is concerned with the question if this specification precisely reflects the informal requirements. This contribution presents an approach for validation of process specifications.

The specification language is given by Petri nets, their behavior by partially ordered process nets (in the sense of Goltz and Reisig). This choice is motivated by experiences within two projects. One of these projects is concerned with high level Petri net models of information systems and business processes (a joint project with researchers from the information system area). The other project studies models of productions systems, given by modular Petri nets equipped with read arcs and event arcs for asymmetric synchronisation (a joint project with reserachers from engineering science).

The core idea of the approach is the generation and visualisation of process nets from a system model given by a Petri net. Advantages compared with traditional simulation are an improved efficiency and expressiveness. In particular, it turned out that a good visualisation of process nets is very close to the users way of thinking about behavior. Algorithmic issues of this approach concern progress and fairness during the construction of processes, selection of alternatives and termination conditions. Adopting the idea of cut-off transitions to depth-first construction of processes and branching processes leads to an efficient termination condition without loosing any information about behavioral properties.

Further features of the approach are and visualization algorithms, an on-the-fly check of certain desirable or undesirable properties (process nets that do not satisfy such properties are ruled out), a graphical language to formulate these properties and the evaluation of processes with respect to cost and time.

## Timed Sequence Diagrams and Tool-Based Analysis – A Case Study

*Karsten Diethers*
TU Braunschweig
Institut für Software

The development of real-time systems remains a complex and errorprone task even if object-oriented concepts are used. Therefore one is interested in validation techniques which allow to analyse the behaviours of a design already in early phases. We use UML timed Sequence Diagrams to specify the real-time behaviour of a communication protocol of audio/video components. The Sequence Diagrams build the requirements specification against which an implementation of the protocol developed by the Bang & Olufsen company is proven correct.

To obtain a complete requirements specification, we have to mark the UML Sequence Diagrams as *optional* or *mandatory* behaviour. Then the Sequence Diagram interactions with their timing constraints and periods are transferred to a setting of timed automata. We use the UPPAAL tool for verification. In particular, we show that the implementation of the protocol conforms to the Sequence Diagram specification concerning the correct data transfer on the bus.

Joint work with Thomas Firley, Michaela Huhn, Thomas Gehrke and Ursula Goltz.

---

## Semantic Formalization of Emerging Dynamics of Compositional Agent Systems

*Pascal van Eck*
Universiteit Twente
Faculteit der Informatica

Agents behaviour is dynamic: both external (e.g., actions, communication) and internal processes (e.g., mental processes) entail continual change. This talk develops a formal semantic model of both external and internal dynamics of agent systems. The two basic concepts used are locality and temporal compatibility. The behaviour of an agent or a component is described by local traces of consecutive states in a specific frame of time. A behaviour of a compositional agent system is described by a combination of local traces of the agents and components within the compositional agent system. Interaction between agents or components (defined by temporal compatibility relations) constrains the possible combinations of traces that may model behaviour of the overall system: only combinations of traces that respect compatibility are allowed.

Joint work with Frances Brazier and Jan Treur.

## Checking Object Systems

*Hans-Dieter Ehrich*
TU Braunschweig
Institut für Software

The question is discussed how global conditions for object systems (i.e., concurrent systems of sequential objects) can be split and assigned to a group of objects such that these objects together satisfy the global condition, provided that each object satisfies its local condition and an appropriate communication mechanism has been established.

The approach is to give the "global" statements in the distributed logic $D_1$ which allows for local assertions about (the states of) other objects in a local logic. A recent result (Ehrich and Caleiro, Acta Informatica 2000) shows that $D_1$ formulae may be translated to $D_0$ in a sound and complete way. $D_0$ is a distributed logic that allows for communication via synchronous action calling. In some cases, the translation can be exploited to split "global" $D_1$ statements to a set of local $D_0$ statements, giving the required communication for free. Alternatively, the communication structure generated by the translation can be checked against the communication mechanism implemented by hand.

---

## Object-Oriented Modeling of Multimedia Applications

*Gregor Engels*
Universität Paderborn
FB 17 - Mathematik / Informatik

The object-oriented modeling language UML (Unified Modeling Language) has been standardized by the OMG (Object Management Group) in 1997. UML was designed as general-purpose language and thus offers a great diversity of modeling constructs. To become applicable in a certain domain, UML has to be adapted by defining so-called prefaces or profiles.

In the talk, ongoing work is presented about such a profile for modeling multimedia applications. The profile is termed OMMMA-L (Object-oriented Modeling of MultiMedia-Appications - the Language). Specific features of OMMMA-L are concepts to model temporal as well as spatial relationships between media objects as, for instance, text, audio, or video. Herefore, UML has been extended by a further diagram type, so-called presentation diagrams, to model the spatial aspect, i.e., the layout of a multimedia application. In order to describe the temporal aspect, i.e., the synchronisation of media objects, sequence diagrams have been extended.

Joint work with Stefan Sauer and Jens Gaulke.

## Message Sequence Charts form the Partially Monoids Point of View

*Piotr Kosiuczenko*
Ludwig-Maximilians-Universität München
Institut für Informatik

Message Sequence Charts (MSC) is a trace language for graphical specification of communication behaviour of asynchronous system components and their environment by means of message interchange. MSC posses a standard semantics based on a new process algebra. It is carefully designed, extensive, operational, but rather complicated. In my talk I have presented the relation of the standard semantics to the theory of traces and partially commutative monoids. I have proved that the MSC language is equivalent to a dependence relation extended by a set of cliques covering the relation. I have extended the standard MSC semantics by allowing synchronous actions and showed that the semantics is closely related to partially commutative monoids. The trace based semantics is in a sense bisimilar with the operational semantics. Presented trace semantics allows to apply the rich theory of partially commutative monoids with associated algorithms and tools. In particular, I have shown that compositions of consistent specifications correspond to subdirect products.

---

## Media – A Model of Distributed Information Systems

*Ulrike Lechner*
Universität St. Gallen
Institut für Medien- und Kommunikationsmanagement

Media are explored as model to envision, to design and to formalize platforms for communities of collaborating human or artificial agents. Seminal to our approach are the media concept and the media reference model. The media concept envisions media as platforms of multi-agent systems and the media reference model determines the main components of a medium and guides its application, e.g., in ECommerce or Knowledge Management. The media concept is formalized to yield a media structure; the formalization of the media reference model yields media requirements. The media structure is a general framework according to which media are modelled. The media requirements capture those notions and concepts that are common to all media and their models. The formaliizatios are based on General Logic, Labelled Deductive Systems and Rewriting Logic. The goal of the formalization is to facilitate artificial agents to reason about media and to act on media autonomously in performing transactions. The goal in providing a framework of metaphors and the formalization is to facilitate human and artificial agents about media and the communities on them in a similar way, such that eventually frictionless communication and collaboration becomes feasible.

Joint work with Beat Schmid and Martina Klose.

## Mobile Maude

*José Meseguer*
SRI International
Computer Science Laboratory

Mobile Maude is a Mobile Agent language extending the rewriting logic language Maude and supporting mobile computation. Mobile Maude uses reflection to obtain a simple and general declarative mobile language design and makes possible strong assurances of mobile agent behavior. The two key notions are *processes* and *mobile objects*. Processes are located computational environments where mobile objects can reside. Mobile objects have their own code, can move between different processes in different locations, and can communicate asynchronously with each other by means of messages. Mobile Maude's key novel characteristics include: (1) reflection as a way of endowing mobile objects with "higher-order" capabilities; (2) object-orientation and asynchronous message passing; (3) a high-performance implementation of the underlying Maude basis; (4) a simple semantics without loss in the expressive power of application code; and (5) security mechanisms supporting authentication, secure message passing, and secure object mobility. Mobile Maude has been specified and prototyped in Maude. A paper, examples, and a simulator written in Maude can be found in `http://maude.csl.sri.com`.

Joint work with Francisco Durán, Steven Eker, and Patrick Lincoln.

---

## Bisimilarity Congruences for Open Terms and Term Graphs via Tile Logic

*Ugo Montanari*
Universit degli Studi di Pisa
Dipartimento di Informatica

The definition of SOS formats ensuring that bisimilarity on closed terms is a congruence has received much attention in the last two decades. For dealing with open system specifications, the congruence is usually lifted from closed terms to open terms by instantiating the free variables in all possible ways, the only alternative considered in the literature relying on Larsen and Xinxin's context systems and Rensink's conditional transition systems. We propose a different approach based on tile logic, where both closed and open terms are managed analogously. In particular, we analyze the 'bisimilarity as congruence' property for several tile formats that accomplish different concepts of subterm sharing.

Joint work with Roberto Bruni, David de Frutos-Escrig and Narciso Marti'-Oliet.

## Model Checking of Causal and Distributed Logics

*Peter Niebert*
Verimag-Centre Equation

Most temporal logics (LTL,CTL,...) used for specification and automatic verification refer to a ”sequential” semantics (sequences of transitions, computation trees, games). The systems verified however usually exhibit a distributed and inherently parallel execution discipline.

In this talk we summerize recent developments in the area of automatic verification of logics referring to the distribution of the system. These logics refer to the causal order of events happening in a distributed system, either referring to infinite partial order executions (in the linear time framework) or to the prefix ordering of finite partial order executions. Examples of such logics include a logic of gossip and an extension of LTL by causal knowledge modalities, both of interest for the specification of distributed algorithms. Automatic verification for such logics has only recently been explored. We propose a framework based on McMillan unfoldings for this purpose and obtain algorithms elementary in the size of the verified system (but in no way efficient). In contrast, model checking for classic logics of knowledge and time are inherently non-elementary in the size of the system.

---

## Dangers of the Past

*Mogens Nielson*
BRICS
University of Aarhus, DK

Applying the open map approach of Joyal, Nielson, Winskel to models with independence (like labelled elementary net systems) with pomsets as observations, results in the so-called hereditary bisimulation. This bisimulation is characterized e.g. by

- an extension of the standard two-player game for standard bisimulation, extended with backwards (past!) moves, and played on the unfolded structures.
- a logic in the form of an extension of the standard Hennessy-Milner logic with backwards (past!) modelities, interpreted over the unfolded structures.

We show that

- bisimularity ist undecidable for labelled finite elementary net systems.
- model checking with standard fix-point operators added over labelled finite elementary net systems is undecidable.

Joint work with Marcin Jurdzinski.

# Translation Validation for Optimizing Compilers

*Amir Pnueli*
Weizmann Institute
Dept. of Appl. Mathematics & Computer Science

The ultimate goal of this work is to guarantee the correct functioning of a compiler. This goal is of particular significance in safety-critical applications and also in *compilation into silicon.* In both cases, it is important that whatever integrity and correctness which have been achieved on the higher (source) level is not damaged by the translation.
Rather than verify the *translator* itself, we choose to verify the results of each *translation.* The advantages of this approach are:

1. Evolution of the translator is not frozen after verification.
2. It is cheaper than translator verification.
3. It is oblivious to the assumptions made by the compiler writers.

In a previous project: SACRES, we constructed a *code validation tool* CVT which could be invoked after every run of the code generator and confirm the correctness of the current run. Applied to a source of 2500 lines and a corresponding target of 4800 lines, it confirmed the translation in 12 minutes. The assumptions that made this possible were:

1. The source and target had each a single external loop with a loop-less straight line (possibly branching) codes as its body.
2. The compiler did not modify any of the arithmetical operations. This enabled us to replace arithmetic operations by uninterpreted functions and derive a special decision procedure to deal with the resulting verification conditions.

In the current work, we plan to extend the translation validation approach, applying it to unrestricted source programs and highly optimizing compilers. As our case study we take the *TRIMARAN* compiler, translating C programs into machine-specific optimized code for a parameterized architectures.

Assume a source program $S$ and its translated target $T$, both presented as transition systems $S =< V^S, \Theta^S, \varrho^S >$ and $T =< V^T, \Theta^T, \varrho^T >$. Our proof method assumes:

1. A *control abstraction* $\kappa$, mapping locations in $T$ into locations of $S$.
2. A *data abstration* $\alpha : (v_1 = E_1) \wedge ... \wedge (v_n = E_n)$, assigning to each source state variable $v_i \in V_i^S - \{\pi\}$ a target expression $E_i$ over the target state variables.
3. For each basic blocks $B_i$ and $B_j$ in $T$, form the verification condition
   $C_{ij} : \alpha \wedge \delta_{ij}^T \wedge \alpha' \rightarrow \delta_{\kappa(i),\kappa(j)}^S$,where $\delta_{ij}^T, \delta_{kl}^S$ are the data components of the respective transition relations.
4. Establish the validity of all generated verification conditions.

It can be shown that this proof method is *sound* and, if completed successfully, establishes that $T$ correctly translated $S$.

When we consider optimized translations, this method is not adequate. For example, an optimized translation of
$l_0 : n = 4; l_1 : x = n + 1; l_2 :$ could yield
$B_0 : n = 4; B_1 : x = 5; B_2 :$

The verification conditions $C_{1,2}$ is no longer valid. The solution to this problem is to *instrument* the compiler to produce a code annotated by invariants. In the above example, we would like to get
$B_0 : n = 4; B_1 : \{n = 4\}\ x = 5; B_2$.
Then , we extend the verification condition to read
$C_{ij} : \varphi_i \wedge \alpha \wedge \varrho^T_{ij} \wedge \alpha' \rightarrow \varrho^S_{\kappa(i),\kappa(j)} \wedge \varphi'_j$.

The method therefore consists of instrumentation of the compiler together with the construction of the code validation tool. In future work we hope to demonstrate the feasibility of this approach.

## Labelled Transition Systems for Object System Behaviour

*Arend Rensink*
Universiteit Twente
Faculteit der Informatica

In this talk we consider the use of labelled transition systems for the purpose of specifying and describing the behaviour of object-based systems. Two considerations play a major role:

- The transitions and their labels should be based on the object's methods;
- Method executions should be modelled as atomic (single-step) transitions as far as possible.

Under those constraints, a single object is usually not an appropriate unit of behaviour: the effect of a method is usually not limited to the object executing it. There are basically two ways around this:

- One may model the communication with the other affected objects explicitly, at the cost of losing atomicity;
- One may take the affected objects into the model, at the cost of losing decomposability.

We consider the second approach, which we call the *closed virtual machine view*, a good basis for the specification of object system behaviour. It gives rise to transition systems which are labelled by one of the following:

- Method invocations on the side of the environment;
- Atomic method executions on the side of the system;
- Non-atomic method starts and method terminations on the side of the system.

Subsequently, we introduce a corresponding *open* virtual machine view, in which the system may itself invoke methods on others in the course of executing its own, and an *active* virtual machine view, in which the system can also undertake autonomous action (by calling other objects' methods). The latter two serve to provide a compositional model underlying the closed system view.

## Basic Principles of the Concurrency-Verification Book

Authors: Willem-Paul de Roever, Frank de Boer, Ulrich Hannemann, Jozef Hooman, Yassine Lakhnech, Mannes Poel and Job Zwiers.

*Willem Paul de Roever*
Universität Kiel
Inst. für Informatik und Prakt. Mathematik

1. Its aim: Formulation of provably sound+complete state-based verification methods for concurrency.

   No fairness, this is left to Amir Pnueli c.s..
   No problem, because 95% of the reasoning about concurrency involves invariance and termination proofs; and because intro of real-time in Vol. II covers most of the remaining need for reasoning about fairness.

2. Methods used:

   - Floyd's Inductive Assertion Method for partial and total correctness
   - Compositional and noncompositional formalisms
   - Hoare logics
   - Communication-closed-layers laws for partial-order-based reasoning, required for reasoning about network protocols.

3. manuscript version of +/- 1993:

   - *Semantics*
     - operational sem. of various kinds: for reasoning about transition diagrams,
     - SoS semantics for reasoning about programs (using program constructs as labels),
     - denotational semantics for compositional reasoning,
     - partial-order semantics,

     which were by-and-large unrelated.
   - *Proof methods*: e.g., a version of the Owicki&Gries method adapted to the inductive assertion framework, and another version of this proof method based on Hoare logic (the original version), which were, again, entirely unrelated.. This resulted in a lot of redundancy/unnecessary overhead in soundness proofs.
   - *Conclusion*
     - Too many unconnected heterogeneous parts.
     - Try to find a uniform basis for concurrent program proving.

4. As uniform basis we choose:

   - Inductive assertion method applied to transition diagrams.
   - Compositionality as leading paradigm.

5. Uniform basis (1): So the iam. became more important.

   - Advantage: soundness and completeness proofs for the iam rather simple.
   - Disadvantage: iam is noncompositional. However, in 1994 Frank de Boer extended the iam to (semantically-based) compositional reasoning about concurrency by integrating Hoare-style rules within the iam, which overcame that obstacle.
   - Compositionality is now introduced when proving completeness of the non-compositional Owicki&Gries and Apt,Francez&de Roever proof methods, by basing these proofs on compositional semantics for shared-variable concurrency and message passing (e.g., reactive sequences, reactive-event sequences, traces, Aczel traces etc., all introduced through simple SoS-like rules).
   - In Hoare Logic, the meaning of syntactically-formulated program constructs now defined by the meaning of their associated transition diagrams, and proof outlines as syntactic abbreviations of inductive assertion networks.
   - As a consequence soundness and completeness proofs for Hoare logic are now reduced to those for the iam, using Cook-style expressability results.

6. Uniform basis (2): No partial-order semantics introduced.
   Instead we introduce a past-time fragment (TTL) of linear-time TL to formulate precedences among actions. And these precedences are then used to define communication-closed layers used in the formulation of CCL laws.

7. Uniform basis (3): Compositionality as a leading theme:

   - Compositionality applies to top-down reasoning.
   - Modularity applies to bottom-up reasoning.

   Both are equally important in practice.

   However, by a result of [Zwiers 89],

   Modularity = Compositionality + Adaptation

   so compositionality remains the central concept.
   Answer to Lamport's critique of compositionality:

   - He applies the concept in his own papers without explicitly defining it.
   - The fact that correctness proofs do not always break up naturally into parts for syntactically separate processes is covered by introducing CCL laws (needed for giving correctness proofs for network protocols).

8. Assume-Guarantee paradigm:

   Compositional proof methods are mainly Assume-Guarantee frameworks for reasoning about open systems:

   - A/C (Misra&Chandy81) for synchronous communication, and
   - R/G (Jones81/83, Starke85) for shared-variable concurrency.

   Problems:

   - No satisfactory completeness proofs published for R/G (and no complete A-G rules as yet available for TL-based specs!).
   - A semantic analysis revealed that such proofs CANNOT be reduced to completeness of the Owicki&Gries method, contrary to prevailing published opinion, because their underlying semantics are different (reactive sequences for O&G, Aczel traces for R/G).
     This took us one year to repair (thesis Ulrich Hannemann/ paper by de Boer,Hannemann & deR in FM'99).

9. All A/G paradigms in the book are now formulated in a semantic setting of:

   - So-called concurrent systems which are composed from transition diagrams (as atoms) using ";" and "//" as operations.
   - Their proofs have a similar structure, using the new concept of A-G-inductive assertion networks for proving transition diagrams compositionally correct, and Hoare-style proofrules for reasoning compositionally about ";" and "//".

## Modeling, Verification and Analysis of Networks of Real-Time Systems

*Jürgen Ruf*
Universität Tübingen
Wilhelm-Schickard-Institut für Informatik

In this talk I present a new modeling formalism that is well suited for modeling real-time systems in different application areas and on various levels of abstraction. These I/O-interval structures extend interval structures by a new communication method, where input sensitive transitions are introduced. The transitions can be labeled with time intervals as well as with communication variables. For interval structures, efficient model checking and analysis techniques based on MTBDDs exist. Thus, after composing networks of I/O-interval structures, efficient model checking and analysis of interval structures is applicable. The usefulness of the new approach is demonstrated by various real-world case studies, including experimental results.

## Evolving Object Specifications for Adaptable Systems

*Gunter Saake*
Universität Magdeburg
Institut für Technische und Betriebliche Informationssysteme

Information systems build the information infrastructure of modern companies and organizations. Information system objects tend to have a very long lifespan - in many cases longer than the lifetime of the corresponding management software. Therefore, information system objects are subject to frequently occurring modifications of behaviour rules. The talk discusses a proposal for extending object specification techniques (based on the TROLL language) with capabilities to supporting evolving behaviour descriptions. The semantics of these added language constructs can be expressed using an extended temporal logic framework. Alternatively, an operational semantics based on concurrent term rewriting using reflection capabilities seems to be feasible.

## Formal Modeling of Operating Systems Concepts – A Methodical Approach

*Katharina Spies*
TU München
Institut für Informatik

To support a systematic approach formal methods like Focus must offer a formal framework for topdown-development of distributed, reactive systems. Focus provides a method "in the large", including special semantics, description techniques and a refinement concept based on message streams.

Formal system development usually starts with a precise representation of the abstract, often informal system requirements. Because of its criticality for further development this first specification must be thoroughly defined. Therefore Focus offers methodical support "in the small": hints and guidelines for constructing formally correct descriptions using special techniques and for systematic modeling in application fields. We present such a methodical approach "in the small" applied to the modeling of operating systems concepts.

Operating systems, seen as ressource manager of a computer system, are modeled as a complex system with lots of interacting components. The consequent use of specially defined specification schemes (translating textual schemes into Focus-formula) leads to formal specifications in a uniform presentation with fixed notation. Further methodical guidelines show how systems can be developed by stepwise adding further behaviour and functioning to the starting core specification. This incremental specification development of an abstract operating system starts with core functioning (scheduler/process management) and leads through virtual memory and interprocess communication to a system with the possibility to create and terminate processes. Using this systematic development approach already defined specifications are extended, adjusted or completely retained.

## Specification and Development of Information Services for the Internet Cable Nets

*Bernhard Thalheim*
Brandenburg University of Technology at Cottbus
Institut für Informatik

Internet Service Sites are under development everywhere; thus making the internet more and more chaotic and unusable. In order to develop sites which are easy to use in an intuitive manner (we call it grandmother-usability), which support the user during solutions of tasks and which are consistent with the intentions and requirements of the owner of the site, we use the codesign approach based on the abstraction layer strategy model. Thus site development is based on consistent development of structure, behavior and interaction. The codesign framework is extended for information-intensive internet sites to codesign of structures, functionality (with corresponding integrity constraints), generalized views (so-called information units) and dialogues. The abstraction layer model has been developed in order to support specification of applications on different abstraction and in different granularity in a consistent manner. Abstraction layers are: motivation layer, requirements analysis layer, business user layer. The theory behind the abstraction layer approach to codesign of information-intensive internet and cable net information service sites is:

- extended ER models (for specification of structure and functionality with corresponding integrity constraints and for specification of generalized views),
- deontic predicate logic (for specification of semantics),
- dialogue objects (for specification of stories, scenarios and scripts),
- process algebra (for computation),
- interaction machines (for computation of interaction),
- intentional logics (for weak semantics of stories),
- tuple space algebra (for presentation objects).

This methodological and theoretical background has enabled us in consistent and well-based development and implementation of internet and cable net information services. The overall size of service for 18 towns and 5 regions is estimated by about 25.000 internet pages. The services are easy to maintain and are displayed simultaneously through internet on one side and through the fiber cable net of the Cottbus area to TV on the basis of videotext and information service based on set-top box technology on the other side.

## Issues in the Semantics of UML Activity Diagrams

*Roel Wieringa*
Universiteit Twente
Faculteit der Informatica

We discuss all elements of UML 1.3 activity diagrams construed as a means to represent workflows. Swimlanes turn out to be a fuzzy construct that is best modelled separately as allocation of activities to actors. Object flows contain many ambiguities that can be removed by having a separate object flow model. Comparison with statecharts reveals that we need wait states, in which the workflow is waiting for some external event to happen. A special kind of wait state is the queue state, in which the workflow is waiting for an actor to be available. Finally, we compare activity diagrams with high-level times hierarchical Petri nets. Disregarding swimlanes and object flows, these have a very similar semantics to the activity diagram semantics that seems to be intended in the UML 1.3 documentation. We end with outlining a program of formalization of activity diagrams in terms of clocked labelled Kripke structures, for which we will use ATCTL as property specification language to do model checking.

Joint work with Rik Eshuis.